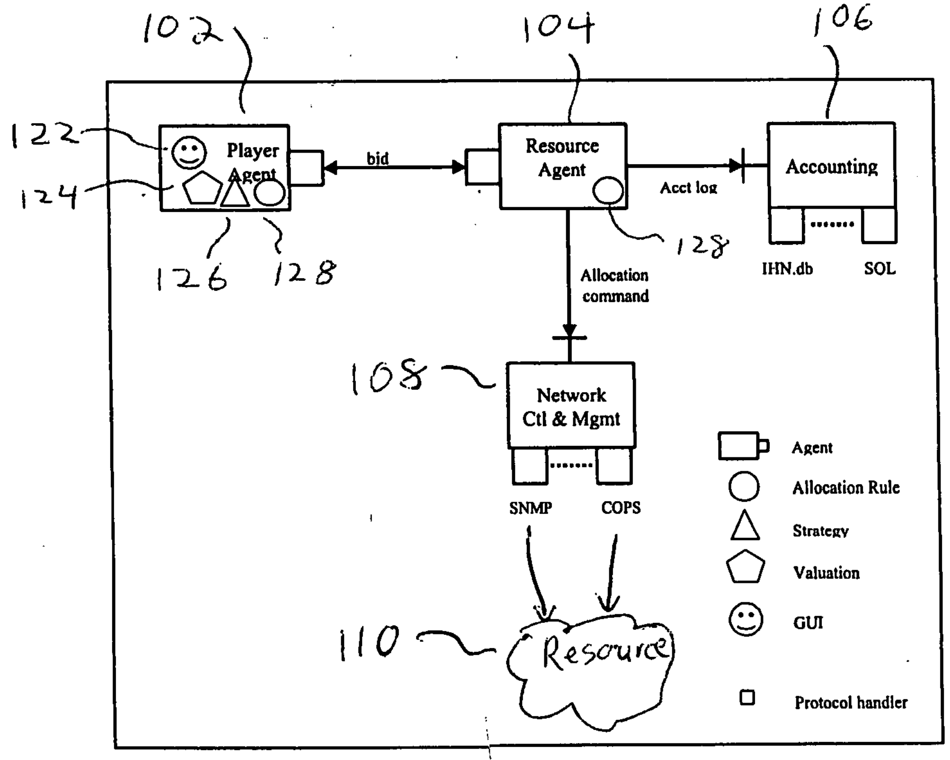

102
104
106
122
124
Player Agent
bid
Resource Agent
Acct log
Accounting
126
128
128
IHN.db
SOL
Allocation command
108
Network Ctl & Mgmt
SNMP
COPS
110
Resource
Agent
Allocation Rule
Strategy
Valuation
GUI
Protocol handler

Fig 1

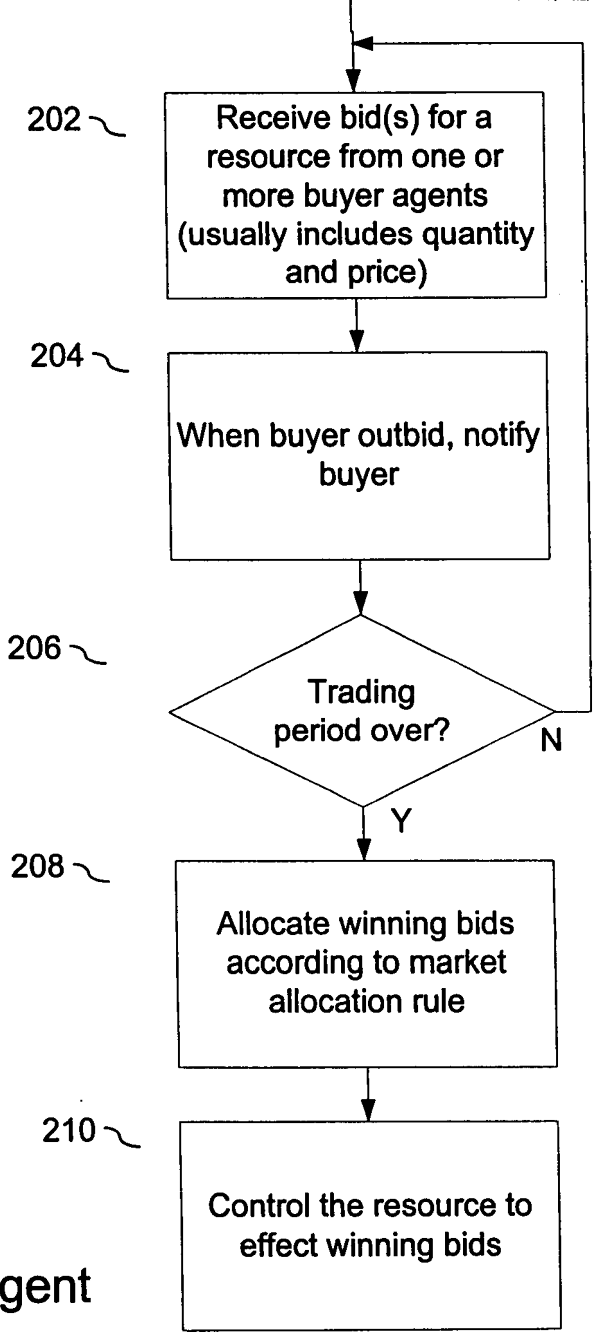
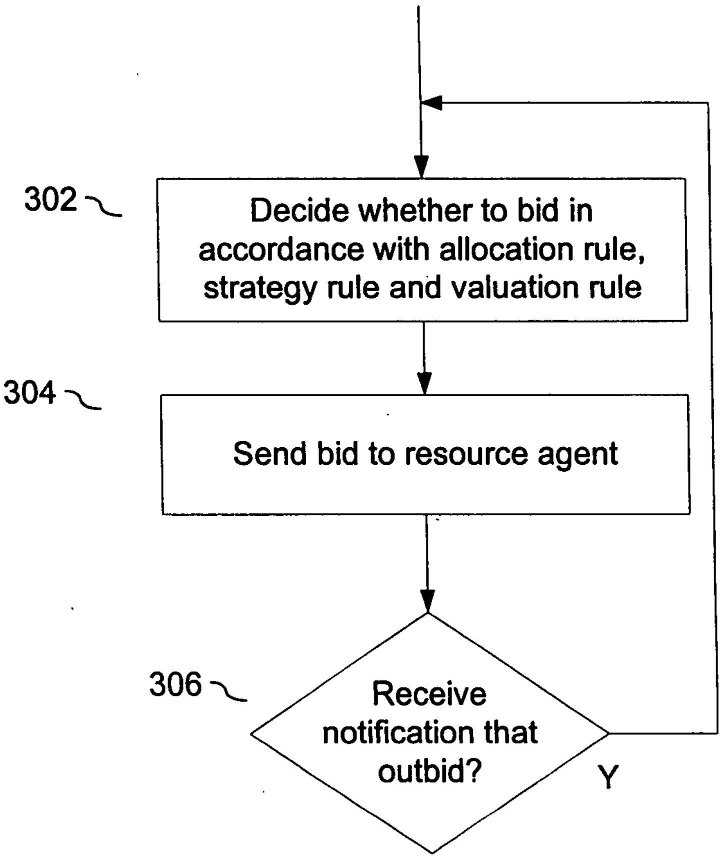

202
Receive bid(s) for a resource from one or more buyer agents (usually includes quantity and price)
204
When buyer outbid, notify buyer
206
Trading period over?
N
Y
208
Allocate winning bids according to market allocation rule
210
Control the resource to effect winning bids

Fig. 2
Resource Agent

Fig. 3
Player agent
(Buyer)


302
Decide whether to bid in accordance with allocation rule, strategy rule and valuation rule
304
Send bid to resource agent
306
Receive notification that outbid?
Y

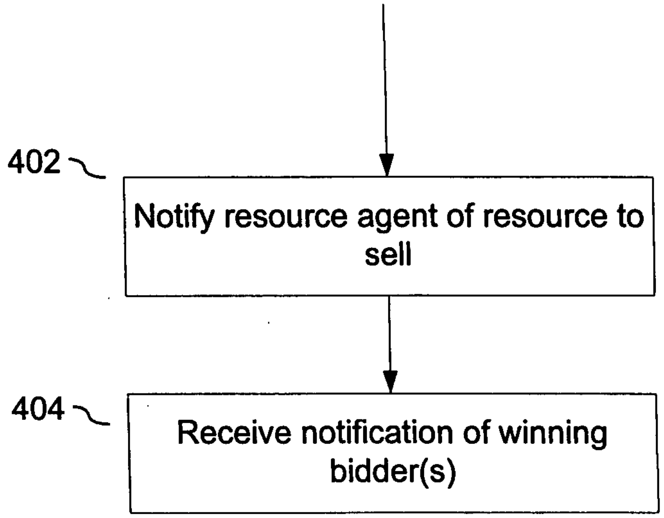

402
Notify resource agent of resource to sell
404
Receive notification of winning bidder(s)

Fig. 4
Player agent
(Seller)

100 units of resource:
A bids for 50 at $3,
B bids for 30 at $2
C bids for 30 at $1
D bid for 20 at $0.50

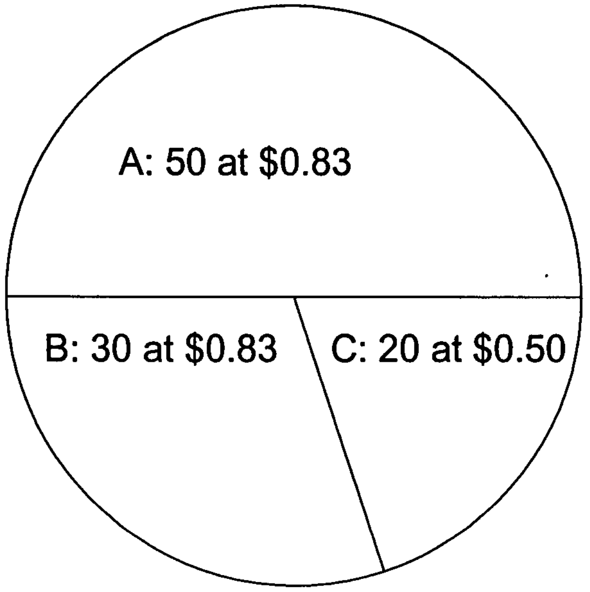


Fig. 5
Example Market Allocation Rule
(PSP)

Quantity	Price	Valuation
-	-	-
-	-	-
-	-	-
-	-	-

Fig. 6(a)
Example
Valuation Rule

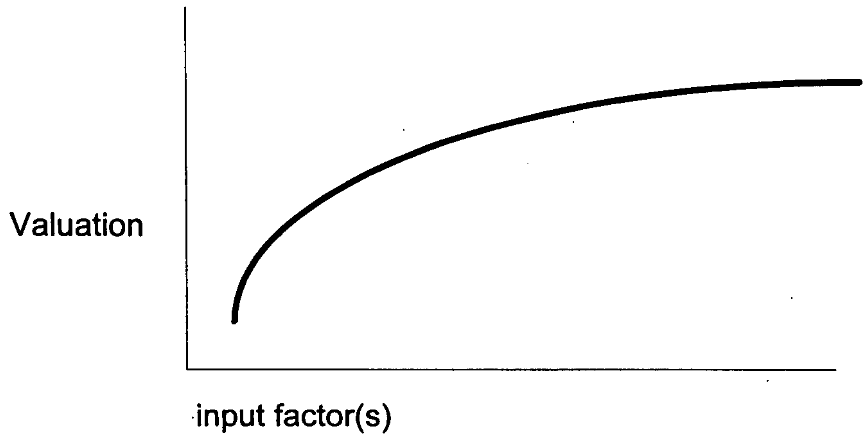


Fig. 6(b)
Example
Valuation Rule

If allocation rule is PSP
[actions for PSP bidding]

If allocation rule is English auction
[actions for English auction]

Fig. 7(a)
Example Strategy
Rule

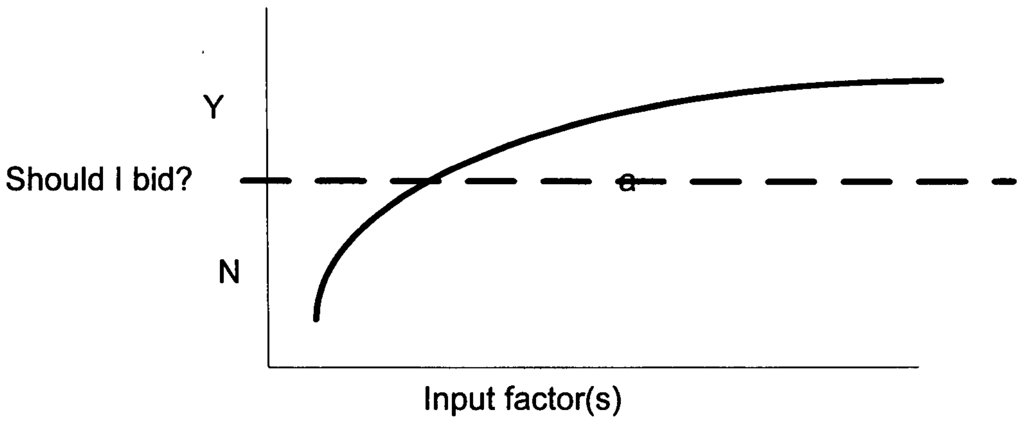


Fig. 7(b)
Example Strategy
Rule

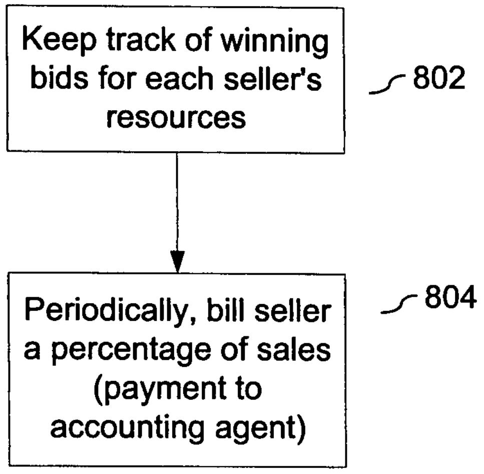

Keep track of winning bids for each seller's resources
802
Periodically, bill seller a percentage of sales (payment to accounting agent)
804

Fig. 8

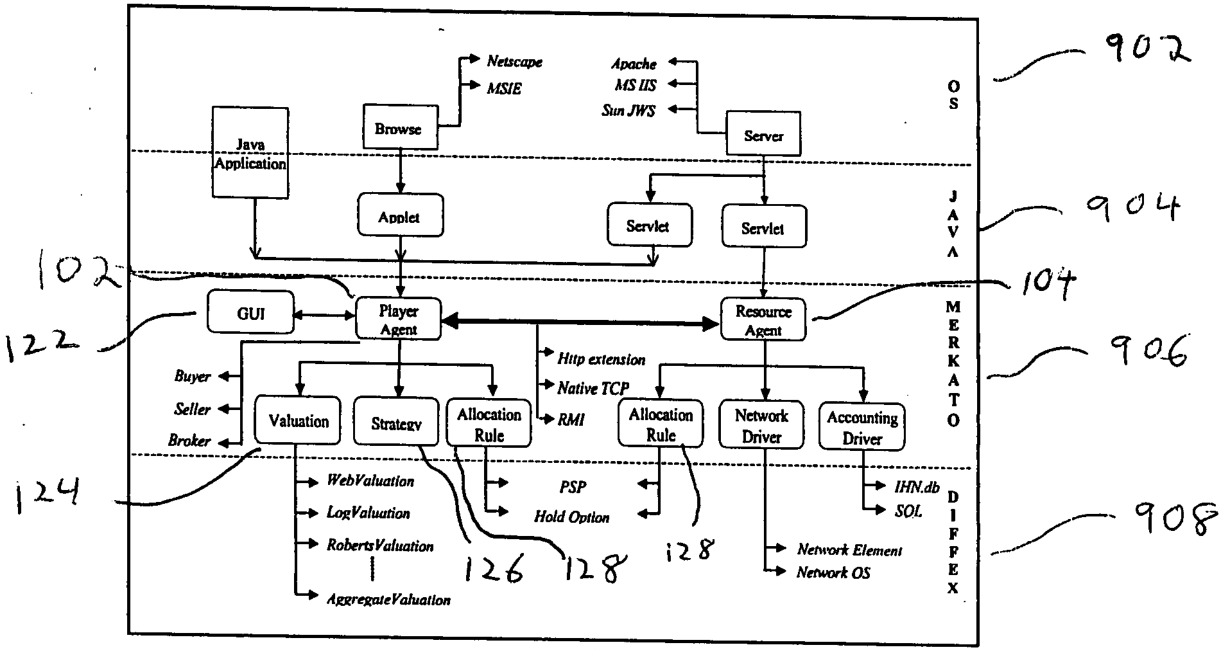
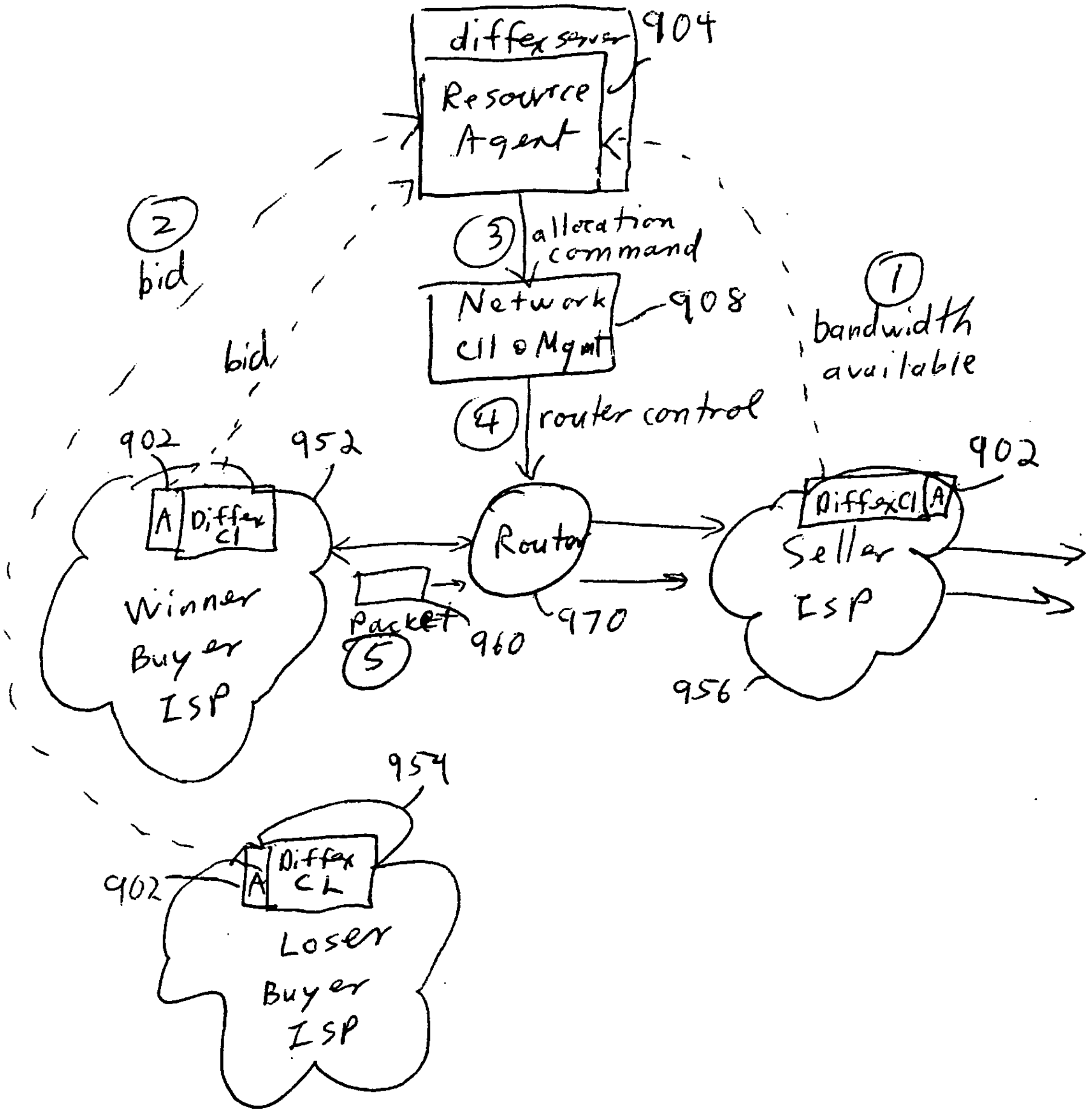

902
904
906
908
O S
J A V A
M E R K A T O
D I F F E X
Java Application
Browse
Netscape
MSIE
Apache
MS IIS
Sun JWS
Server
Applet
Servlet
Servlet
102
104
122
GUI
Player Agent
Resource Agent
Buyer
Seller
Broker
Http extension
Native TCP
RMI
Valuation
Strategy
Allocation Rule
Allocation Rule
Network Driver
Accounting Driver
124
WebValuation
LogValuation
RobertsValuation
AggregateValuation
PSP
Hold Option
126
128
128
IHN.db
SQL
Network Element
Network OS

Fig. 9(a)



diffex server
904
Resource Agent
2
bid
bid
3
allocation command
Network Ctl & Mgmt
908
1
bandwidth available
4
router control
902
952
A
Diffex Cl
902
Diffex Cl
A
Router
Seller ISP
Winner Buyer ISP
Packet
5
960
970
956
954
902
A
Diffex CL
Loser Buyer ISP

Fig 9(b)

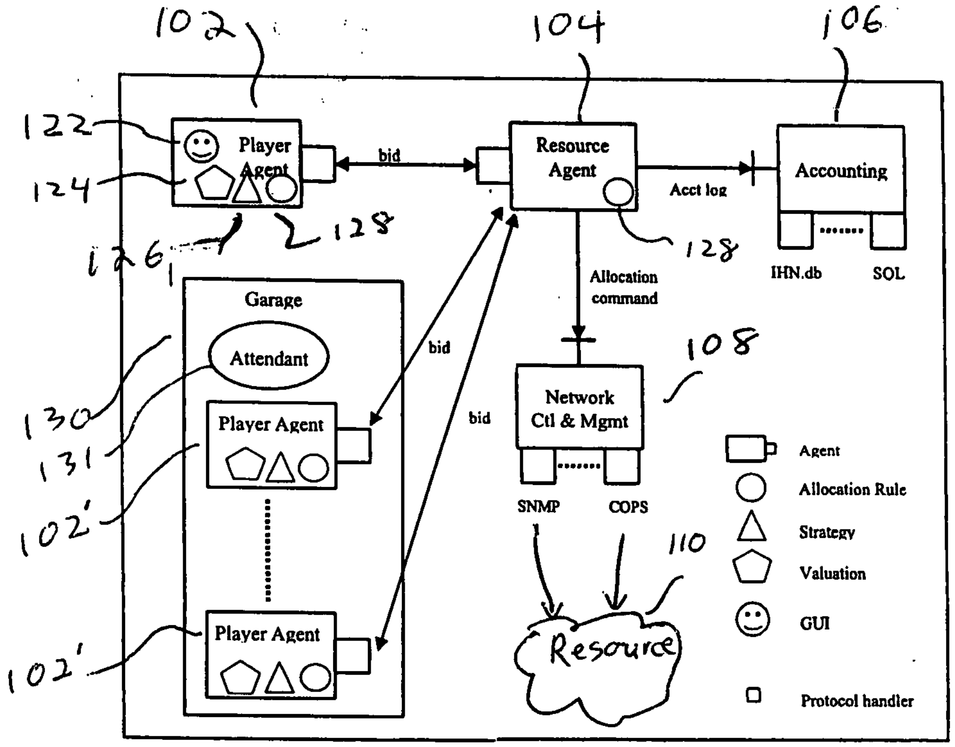

102
104
106
122
124
126
128
Player Agent
bid
Resource Agent
Acct log
Accounting
128
IHN.db
SOL
Allocation command
Garage
Attendant
bid
bid
108
130
131
102'
Player Agent
Network Ctl & Mgmt
SNMP
COPS
110
Resource
102'
Player Agent
Agent
Allocation Rule
Strategy
Valuation
GUI
Protocol handler

Fig 10

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <AuctionPlayer context="http://HOSTNAME:HTTP_PORT/bx/garage">
  - <SingleFrameGUI>
      <TextPanel name="News" height="50" visible="true" border="false" />
      <LoginPanel name="Login" height="160" visible="true" border="true" />
      <ResourceAgentPanel name="ResourceAgent" height="80" visible="true"
        border="true" />
      <UploadAgentPanel name="Garage" height="80" visible="true"
        border="true" />
      <BidCanvasPanel name="BidCanvas" height="180" visible="false"
        border="true" />
    - <StrategyChoicePanel name="Strategies" height="160" visible="true"
        border="true">
        <StrategyPanel name="Manual" strategy="ManualStrategy" />
        <StrategyPanelNotEditable name="Auto" strategy="TruthfulStrategy" />
      </StrategyChoicePanel>
    - <ValuationChoicePanel name="Valuations" height="240" visible="false"
        border="true">
        <WebValuationPanel name="Web Valuation" valuation="WebValuation" />
        <ValuationPanel name="Elastic Demand" valuation="RobertsValuation" />
        <ValuationPanel name="Inelastic Demand" valuation="LinearValuation" />
        <BudgetValuationPanel name="Budget Valuation" label=""
          valuation="BudgetValuation" />
      </ValuationChoicePanel>
      <PlayerInfoPanel name="Allocation" height="120" visible="true"
        border="true" />
      <BudgetPanel name="Budget" height="80" visible="false" border="true" />
      <DisplayPanel name="Units" height="80" visible="false" border="true" />
      <IPAddressPanel name="IP" height="110" visible="false" border="true" />
      <ConnectionPanel name="Connection" height="140" visible="false"
        border="true" />
      <BidTablePanel name="Bid Table" height="400" visible="false"
        border="true" />
      <BidGraphPanel name="Bid Graph" height="400" visible="false"
        border="true" />
      <AllocationGraphPanel name="Allocation Graph" height="400" visible="false"
        border="true" />
    </SingleFrameGUI>
    <PlayerIdentity name="USERNAME" passwd="PASSWD" ipaddress="IP_ADDRESS"
      netmask="NETMASK" />
  - <LinearValuation label="Inelastic Demand">
      <Parameter name="qmax" value="45000.0" label="Kbps" />
      <Parameter name="vmax" value="44928.0" label="$/month" />
    </LinearValuation>
  - <RobertsValuation current="false" label="Elastic Demand">
      <Parameter name="qmax" value="45000.0" label="Kbps" />
      <Parameter name="vmax" value="4928.0" label="$/month" />
    </RobertsValuation>
  - <BudgetValuation current="true" label="Budget Valuation">
      <Parameter name="qmax" value="1000.0" label="Kbps" />
      <Parameter name="budget" value="100.0" label="$/month" />
    </BudgetValuation>
```

11 (a)

```
- <WebValuation label="Web Valuation">
    <param name="delay" value="100.0" />
    <param name="hitspermonth" value="100000.0" />
    <param name="filesize" value="1000.0" />
    <param name="centsperhit" value="0.1" />
    <param name="randomize" value="false" />
  </WebValuation>
  <Parameter name="budget" value="51840.6" label="$/month" />
  <ManualStrategy current="false" label="Manual" />
  <TruthfulStrategy current="true" label="Auto" />
  <resourceAgentURL nickname="RESOURCE_NAME"
    current="true">http://HOSTNAME:HTTP_PORT/bx/RESOURCE_NAME</resourceAge
  <uploadURL nickname="HOSTNAME
    garage">http://HOSTNAME:HTTP_PORT/bx/garage</uploadURL>
  <param name="playerInterval" value="2000" />
  <param name="timeout" value="2000" />
  <param name="timelabel" value="min" />
  <param name="currencylabel" value="c" />
  <param name="quantitylabel" value="Mbps" />
  <param name="debug" value="false" />
</AuctionPlayer>
```

11 (b)

```
/*
 * File:        Truthful.java
 *
 * Remark:      Strategy for player with diminishing returns
 *
 * $Id: Truthful.java,v 1.16           07:43:19 cobe Exp $
 *
 *
*/

package ihn.merkato;
import org.w3c.dom.*;
import com.sun.xml.tree.XmlDocument;

/**
 * The strategy that bids the truthful best reply as in Proposition 1 o
f
 * the PSP paper.
 * It will only submit the bid if utility will be increased by at least
 * epsilon.
 *<p>
 *@author Nemo Semret
 *(
 */
public class Truthful extends AuctionStrategy {

  Bid tmp = createBid();
    /**
   * Finds truthful best reply as in Proposition 1 of
   * the PSP paper.
   * Sets the bid at the player if utility will be increased by at leas
t
   * epsilon.
   *<p>
   * If timelogging is enabled, this will write to the  player's
   * log a line with current time, bid, allocation, and utility,
   * at each call.
   * @see #epsilon
   * @see ihn.merkato.AuctionPlayer#setBid
   */

  public boolean bid() {

    double lq=0, uq= getPlayer().getValuation().qmax(), mq= (uq+lq)/2,
      dq =  getPlayer().dq();

    if(debug()) {
        getPlayer().log("q range =  ["+lq+
                        ","+uq+"] dq="+dq+
                        " Q="+getPlayer().stuff());
        getPlayer().addnews(".");
    }
```

12(a)

Example of Agent strategy

```
    // see Proposition 1
    int i=0;
    double mp, dv;
    while( uq-lq >dq && i<20) {
      i++;
      mq = (lq+uq)/2;
      /*
         if(mq < getPlayer().stuff() -
         (getBidder().getBidList()).demandAtPrice(
         getPlayer().dval(mq, mq+dq),
         getPlayer().getId()))
         */
      // the following is equivalent and more general
      dv=getPlayer().getValuation().dval(mq, mq+dq);
      mp=getBidder().getBidList().marketPrice(getPlayer().stuff()
                                                    -mq,
                                                    getPlayer().getId()
);
      if(debug())
        getPlayer().log("i="+i+" mq="+mq+" dv="+dv+" mp="+mp);

      if(dv>mp)
        lq=mq;
      else
        uq=mq;
    }

    tmp.bidderid = getPlayer().getId();
    tmp.price = Data.MAXPRICE;
    tmp.qty = lq;

    if(debug())
      getPlayer().log(""+i+" steps. q range =  ["+lq
                      +","+uq+"] currentbid="+
                      getBidder().anteBid()+ " found "+tmp);

    if(util(tmp) <0) {
        uq= tmp.qty;
        lq=0;
    }

    i=0;
    while(uq-lq>dq && i<20) {
      tmp.qty = (uq+lq)/2;
      i++;
      if(debug())
        getPlayer().log("i="+i+" q="+tmp.qty);
      if(util(tmp) <0)
        uq= tmp.qty;
      else
        lq = tmp.qty;
```

12(b)
Example of Agent Strategy

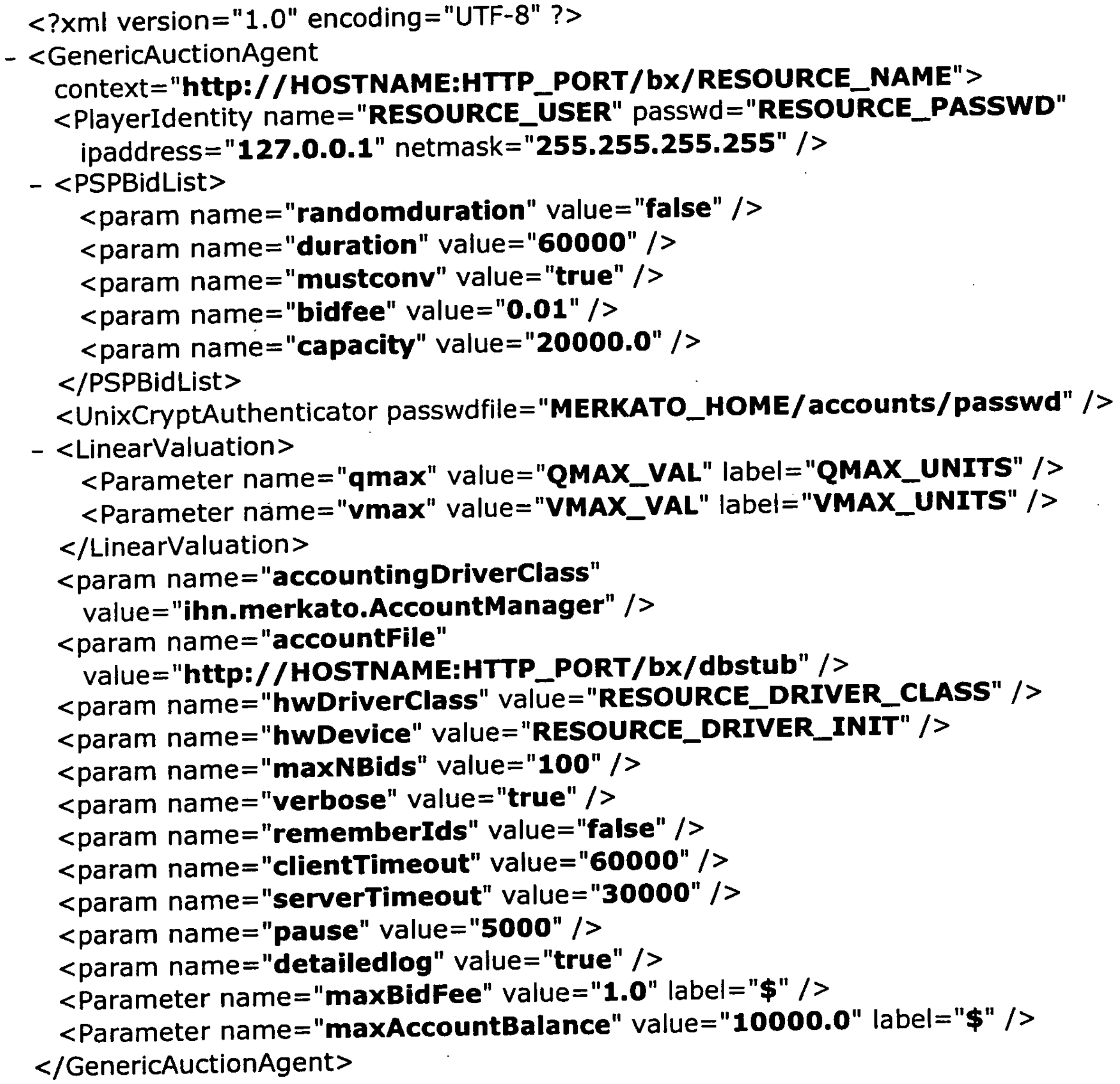
```
    }

    // need this in case the above loop is just outside the budget
    while (util( tmp) <0 && tmp.qty>0 && i <40) {
      i++;
      tmp.qty -=dq;
      if(debug())
        getPlayer().log("i="+i+" q="+tmp.qty);
    }

    if(debug())
      getPlayer().log(""+i+" steps. range=["+lq+","+uq+"] currentbid="+
                           getBidder().anteBid().qty);

1232 tmp.price = getPlayer().getValuation().dval(tmp.qty, tmp.qty+dq);

    double u = getPlayer().currentUtil();
    double newu = util(tmp) ;

    if(debug()) {
      getPlayer().log("currentalloc="+
                           getBidder().currentAllocation()+
                           " newbid="+tmp+" antebid="+
                           getBidder().anteBid());
      getPlayer().log("u="+u+" newu="+newu+" ante="
                       +util(getBidder().anteBid())
                       +" fee="+getBidder().bidFee()
                       +" epsilon="+epsilon()
                       +" avgdur="+getAvgDuration());
    }

    if(getPlayer().trace()) {
      Bid alloc = getBidder().currentAllocation();

      getPlayer().log(""+getBidder().anteBid().qty
                           +"\t"+getBidder().anteBid().price+"\t"+
                           alloc.qty+"\t"+alloc.price+"\t"+
                           getPlayer().currentUtil());
    }

1234 if ( newu > u + epsilon()) {
        if(debug()) getPlayer().addnews("*");
        return getBidder().setBid(tmp.qty, tmp.price);
    }
    else {
        if(debug()) getPlayer().addnews("-");
        return false;
    }
  }
}
```

12(c)
Example of Agent Strategy

```
/*
 *   BidList object
 *
 * File:         PSPBidList.java
 *
 *
 *
 *
 *
 *
 *
*/

package ihn.diffpex;

import ihn.merkato.Bid;
import ihn.merkato.Data;
/**
 *
 *
 */
public class PSPBidList extends ihn.merkato.GenericBidList {

  /**
   * Compute an allocation given the current profile of opponents
in
   * this bidlist. This  class uses the Progressive-Second-Price
   * auction rule.
   * @param tb The bid for which the allocation is to be calculate
d.
   * @param Q  The total quantity of resource available.
   */

  public  Bid allocation(Bid tb, double Q) {
    return PSPallocation(tb, Q);
  }

  /**
   * Compute an allocation given the current profile of opponents
in
   * this bidlist, with the Progressive-Second-Price
   * auction rule.
   * @param tb The bid for which the allocation is to be calculate
```

```
d.
   * @param Q  The total quantity of resource available.
   */

  private  synchronized Bid PSPallocation(Bid tb, double Q) {
    Bid index = top;
    Bid alloc= new Bid();
    double leftover = Q;  //leftover with player id
    double leftoverwo =Q; //leftover without player id
    double qAj,qAj0, num=0, den=0;
    boolean gotcha = false;

    alloc.qty = Math.min(tb.qty,
                         Math.max(Q-demandAtPrice(tb.price, tb.bid
derid),0));

    while(index.next != null) {
      index = index.next;

      if(index.bidderid != tb.bidderid) {

        if(index.price <= tb.price && !gotcha) {
          leftover -= tb.qty;
          if(leftover <=0)leftover=0;
          gotcha = true;
        }

        qAj = (index.qty <= leftover ? index.qty : leftover);
        qAj0= (index.qty <= leftoverwo ? index.qty : leftoverwo);
        num += (index.price* (qAj0- qAj));
        //      den += (qAj0- qAj);

        leftoverwo  -= index.qty;
        leftover -= index.qty;
        if(leftover <=0)leftover=0;
        if(leftoverwo <=0)leftoverwo=0;

      }


    }

    //    if (!gotcha) alloc.qty = (tb.qty <= leftover ? tb.qty :
```

```
leftover);

    //    alloc.price = den>0 ? num/den : 0;

    alloc.price = alloc.qty>0 ? num/alloc.qty : 0;
    alloc.bidderid = tb.bidderid;

    return alloc;
  }

    /**
     * Bids with ID#0 are not counted.
     */
  public double revenue(double Q) {
    Bid index = top;
    double r=0;
    int I=0;
    Bid al;
    while (index.next != null) {
      index = index.next;
      I++;
      if(index.bidderid!=0) {
          al = allocation(index,Q);
          r+= al.qty*al.price;
              //value(index.bidderid,Q);
      }
    }
    return r;
    //    if(I==0) return 0;
    //    else  return r -=  (I-1)*value(Data.NOBODY, Q);
  }


}
// substituted 8 float to double
```



```xml
  <?xml version="1.0" encoding="UTF-8" ?>
- <GenericAuctionAgent
    context="http://HOSTNAME:HTTP_PORT/bx/RESOURCE_NAME">
    <PlayerIdentity name="RESOURCE_USER" passwd="RESOURCE_PASSWD"
      ipaddress="127.0.0.1" netmask="255.255.255.255" />
  - <PSPBidList>
      <param name="randomduration" value="false" />
      <param name="duration" value="60000" />
      <param name="mustconv" value="true" />
      <param name="bidfee" value="0.01" />
      <param name="capacity" value="20000.0" />
    </PSPBidList>
    <UnixCryptAuthenticator passwdfile="MERKATO_HOME/accounts/passwd" />
  - <LinearValuation>
      <Parameter name="qmax" value="QMAX_VAL" label="QMAX_UNITS" />
      <Parameter name="vmax" value="VMAX_VAL" label="VMAX_UNITS" />
    </LinearValuation>
    <param name="accountingDriverClass"
      value="ihn.merkato.AccountManager" />
    <param name="accountFile"
      value="http://HOSTNAME:HTTP_PORT/bx/dbstub" />
    <param name="hwDriverClass" value="RESOURCE_DRIVER_CLASS" />
    <param name="hwDevice" value="RESOURCE_DRIVER_INIT" />
    <param name="maxNBids" value="100" />
    <param name="verbose" value="true" />
    <param name="rememberIds" value="false" />
    <param name="clientTimeout" value="60000" />
    <param name="serverTimeout" value="30000" />
    <param name="pause" value="5000" />
    <param name="detailedlog" value="true" />
    <Parameter name="maxBidFee" value="1.0" label="$" />
    <Parameter name="maxAccountBalance" value="10000.0" label="$" />
  </GenericAuctionAgent>
```

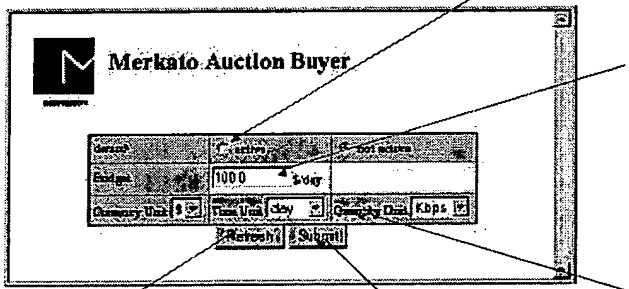
Fig 14

HTML – Not Bidding

Note: All changes require that "Submit" be pressed to send change to agent in "garage"

Select "active" to begin bidding

"Budget" is used to calculate price per unit bandwidth bid during auction. (Must enable "active" first). User is encouraged to bid high during periods of heavy use and bid low, or not at all during periods of light use.



Merkato Auction Buyer

○ active ● not active

Budget 1000 $/day

Currency Unit $ Time Unit day Quantity Unit Kbps

"Refresh" updates screen display

"Submit" sends new values to agent in "garage" and exits

Select the units for the display. Previously entered values will scale to the new units

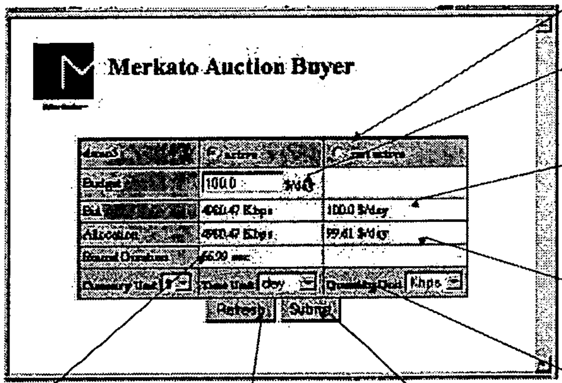
There is no cost accrued to buyers who are not bidding. They will be placed in the best-effort queue until they elect to bid for bandwidth.

"Submit" updates the budget value of the garaged agent to what you have entered into this screen and exits. At this point, it exits to a generic Merkato screen, but for customers, it will exit to a StreamingHand page.

Fig 15(a)

HTML - Bidding

Note: All changes require that "Submit" be pressed to send change to agent in "garage"



Merkato Auction Buyer

	● active	○ not active
Budget	100.0 $/day	
Bid	4960.47 Kbps	100.0 $/day
Allocation	4960.47 Kbps	99.41 $/day
Round Duration	66.29 sec	
Currency Unit $	Time Unit day	Quantity Unit Kbps

Select "Inactive" to stop bidding

Budget (same as in "inactive" screen)

This is the last price offered with quantity desired bid (changes often so need to "Refresh").

Amount of bandwidth and extended price allocated to this agent during the last auction round

Duration of last bid round. This time varies based on how active the bidding is during a round.

Refresh (same in in "inactive" screen)

Submit (same in in "inactive" screen)

Units (same in in "inactive" screen)

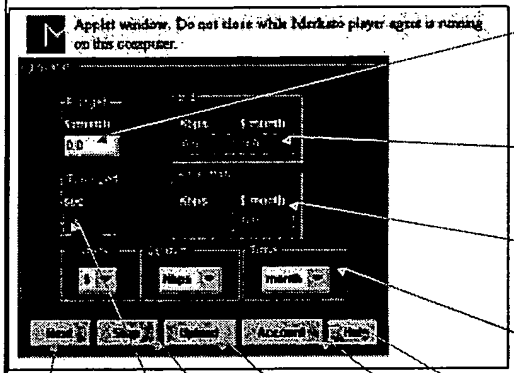
User will generally want to bid high during period of heavy use and lower during periods of light use.

The agent will attempt to obtain as much bandwidth as possible without exceeding budget. Conversely, the agent will request smaller and smaller amounts of bandwidth until they can obtain something for the bud get price.

"Refresh" updates the screen. It does not send any changes to the "budget" value to the garaged agent. "Submit" does this (and then exits).

Fig 15(b)

Wizard – Bid Window


Applet window. Do not close while Markato player agent is running on this computer.
0.0
Mbps
$ month
Next
Stop
Upload
Account
Help

"Budget" is used to calculate price per unit bandwidth bid during auction.

This is the last price offered with quantity desired bid (changes often so need to "Refresh").

Amount of bandwidth and extended price allocated to this agent during the last auction round

Select the units for the display. Previously entered values will scale to the new units

Show auction graph

Stop bidding

Countdown timer for current auction. Reset whenever a new bid is received.

Uploads agent to garage where it can continue bidding and exits

Display account summary screen

Display help. When checked, mouse-button presses bring up help rather than performing function

"Stop" means to stop bidding. This bidding agent will not be charged and they will be placed in the shared best-effort queue.

"Upload" uploads the configuration to the garaged agent. Not that this will change some advanced settings to those assumed by this simple valuation and strategy model.

This simple budget-based valuation model has the bidding agent attempt to get as much bandwidth as possible without exceeding the budget number.

The strategy is based on the formula: price-per-unit-bandwidth * bandwidth-allocated = total-price-paid

Where the total-price-paid ("budget") is held constant, and the other two variables allowed to be altered.

Following this strategy, the bidder will first attempt to get all the bandwidth the seller is offering for their budgeted amount, which works out to the lowest possible price-per-unit-bandwidth. If unsuccessful, the bidding agent gradually increases the offered price-per-unit-bandwidth and decreases the desired amount of bandwidth, until they successfully win an allocation.

If all bidders follow this valuation model, they will each get a bandwidth allocation that is the same proportion to total bandwidth as their budget is to the combined budgets of all bidders.

From the "Help" screen"

- Press **Start** to tell your agent to start bidding for you.
- Press **Stop** to tell your agent to stop.

Fig 15(c)

“Bid Canvas” Screen

A dynamic display of the second price auction in progress

Red circle is the bid you intend to submit.

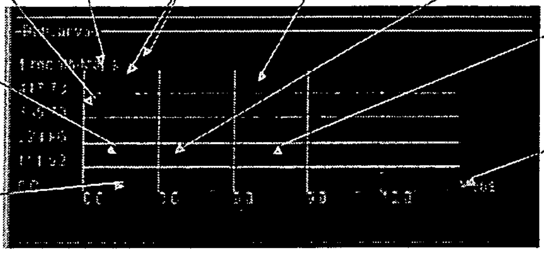
Blue dots are other bidders. Their position represents the quantity of bandwidth and the price per unit bandwidth they offered

Solid Magenta dot is the allocation you received last round

Solid red dot is your last bid. Red number next to dot is your bidder ID number

Magenta line represents your budget. Agent, if automated, will never bid above this line, regardless of bid (valuation) strategy line

Magenta circle is the allocation you would get based on your last bid.



Red line represents bidding (valuation) strategy. Your bids, if automated, will follow this line.

Blue shaded area represents bandwidth allocated, from right to left. If different bidders pay different unit prices, it will look like steps. Black space at left indicates that not all bandwidth was allocated based on the current bids.

Far right dot is seller. Shows price floor and bandwidth being sold

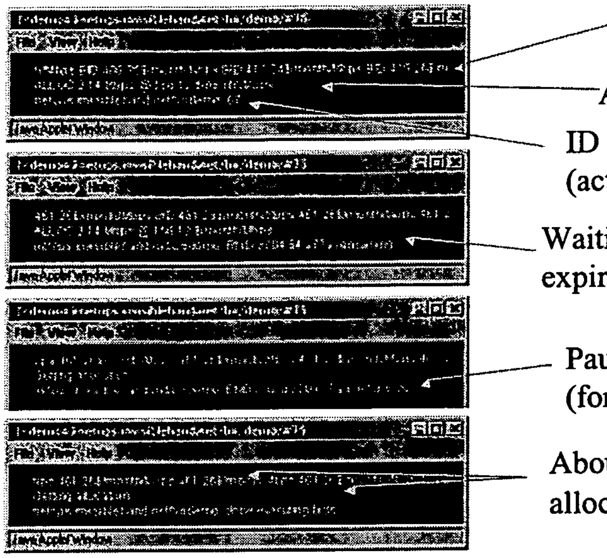
Note: Manual bids may be placed by placing your cursor at the position which represents your bid and clicking your mouse.

Red Valuation Line and Magenta Budget line are superimposed when “Budget” valuation is being used (default for “HTML” and “Wizard” Agent Interfaces).

Often the Red circle, red dot and magenta circle will be very close together.

Fig 15(d)

Status bar



"Tickertape" list of price bids

Allocation last round

ID of last bidder (active auction)

Waiting for bid timer to expire at end of auction

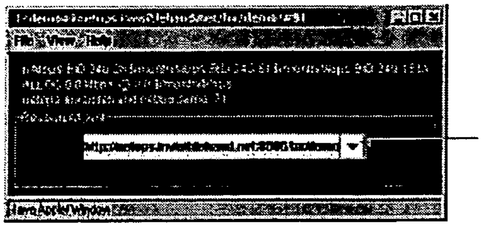
Pause at end of auction (for configuration)

About to receive allocation update

Fig 15(e)

"Resource Agent" Subscreen

Selection screen for resource for which you are bidding



Pull-down menu allows you to determine which resource you would like to bid on.

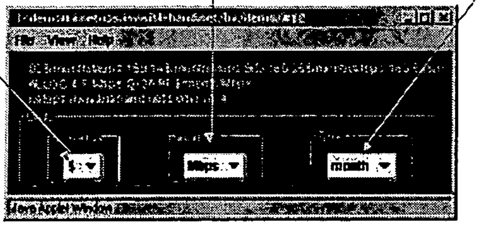
Fig 15(f)

“Units” Subscreen

Enter the units for currency which you would like in all displays (currently, the only option is “$”)

Enter the units for bandwidth which you would like in all displays (options are “Kbps”, “Mbps”, “Gbps”)

Enter the units for time which you would like in all displays (options are “ms”- millisec, “sec”, “min”, “hr” – hours, “day”, “or “month”).



Note: If you change units, any numerical values in any other subscreen will automatically be scaled to reflect the units change, but represent the same quantity as originally specified.

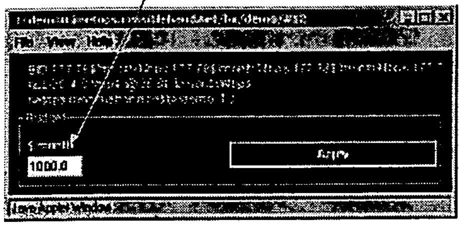
Fig 15(9)

“Budget” subscreen

Selection screen for bidding “strategy”

You enter the “maximum cost run rate” here - this supercedes any higher values that might be derived from valuation curves. In other words, bids - which consist of a price per unit bandwidth and a total bandwidth desired - will not be placed if they would result in a greater price than that indicated here.

When the valuation type is “Budget”, the “price per unit time” field cannot be altered because it is a duplicate of that entered in the valuation subscreen.



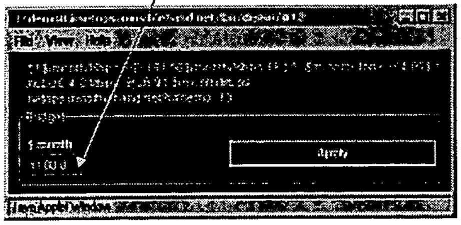
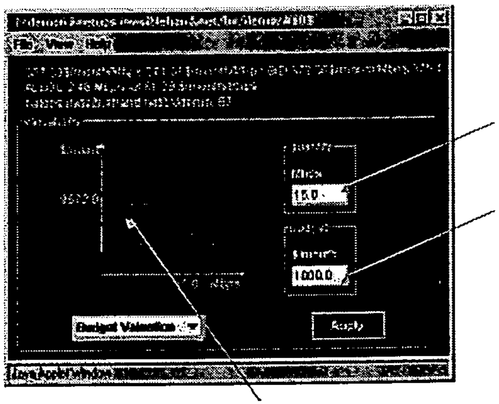


Fig 15(h)

Valuations – Budget Valuation

Selection screen for bidding "strategy"



Maximum quantity of bandwidth desired (by default it is all the seller is offering)

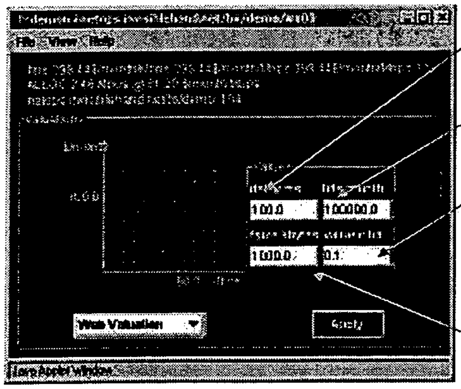
Maximum amount you are willing to pay for that bandwidth

This curve represents the value you place on bandwidth based on the amount you receive. The "budget" valuation curve represents a desire to get the maximum amount of bandwidth for a constant price. The "Valuation" curve in the bid canvas as your bid strategy is derived from the change in slope of this curve.

Fig 15(i)

Valuations – Web Valuation

Selection screen for bidding "strategy"



100.0

100000.0

1000.0

0.1

Average desired delay in ms to transfer a file of the size indicated.

Number of such files expected to be downloaded per month

Value to you in cents per file downloaded

(Note: You can independently set your maximum monthly budget via the "Budget" screen, so the shape of this curve is more important than its maximum price-point)

Average size of file downloaded

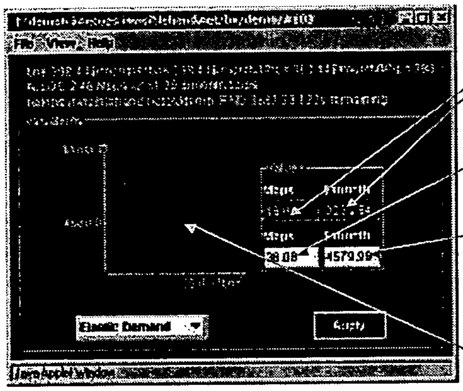
The web valuation attempts to translate a content hosting business need into bandwidth and price requirements. The formulas used are: (max bandwidth in Mbps) = fsize * 8 / delay

(Max price in $/month) = value * (hits per /month) / 100

Fig 15(j)

Valuation – Elastic Demand

Selection screen for bidding "strategy"



This display provides the user a feel for the shape of the curve. The right display is the price-point for the amount of bandwidth to the left.

Max bandwidth desired (see discussion, below, for impact of this setting)

Max price-point (see discussion, below, for impact of this setting)

Note : You can independently set your maximum monthly budget via the "Budget" screen, so the shape of this curve is more important than its maximum price-point

Note 2: You may enter new values by dragging and dropping the red dot on the graph with your cursor

"Elastic" valuation models how users have historically valued internet bandwidth. The formula, when used as a bid strategy (see Bid Canvas), is:

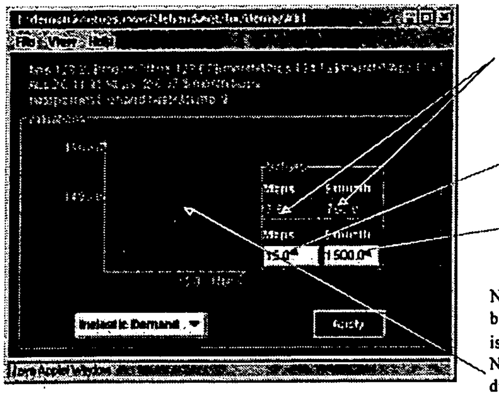
(Price per unit bandwidth) * (Qty of Bandwidth)^2 = (.0012) * (Max price-point) * (Max bandwidth)^2

Note 3: Constant (.0012) is correct for units shown, above. It will scale depending on units selected.

Fig 15(k)

Valuation – Inelastic Demand

Selection screen for bidding "strategy"



This display provides the user a feel for the shape of the curve. The right display is the price-point for the amount of bandwidth to the left.

Max bandwidth desired (see discussion, below, for impact of this setting)

Max price-point (see discussion, below, for impact of this setting)

Note : You can independently set your maximum monthly budget via the "Budget" screen, so the shape of this curve is more important than its maximum price-point

Note 2: You may enter new values by dragging and dropping the red dot on the graph with your cursor

"Inelastic" valuation indicates that you wish to pay the same price per unit bandwidth regardless of the amount of bandwidth received. This results in a horizontal bid strategy line on the Bid canvas, following the formula:

(Price per unit bandwidth) = (Max price-point) / (Maximum Bandwidth Desired)

When the elastic bid strategy is combined with the knowledge of the second price auction mechanism, it results in the following behavior:

If the elastic valuation is above the budget line, the agent will do a reverse calculation to determine when it can bid on the valuation line, but obtain the bandwidth on the budget line.

If the elastic valuation is below the budget line, the agent will continue to ask for the maximum amount of bandwidth at the valuation price and not accept a lesser amount of bandwidth.

Fig 15(1)

"Strategy" Screen

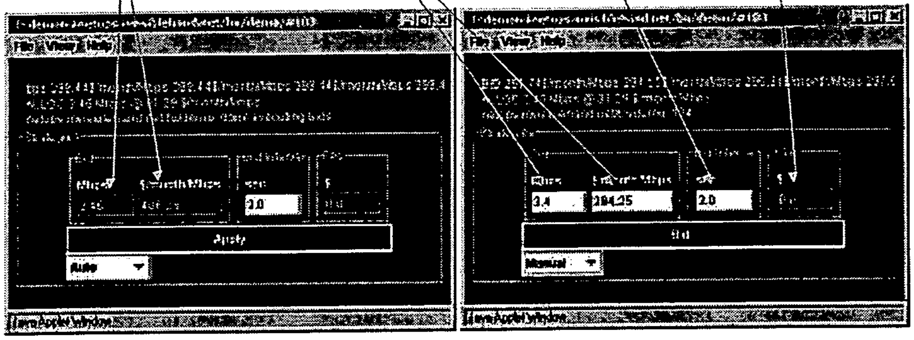
Display of bids being submitted in real time

"Bid" area indicates current bid in automatic mode

"Bid" area allows entry of manual bid in manual mode

Settable time between automated bids

Display of bid fee (Bid fee is an optional feature to prevent frivolous bidding)



3.0

3.4

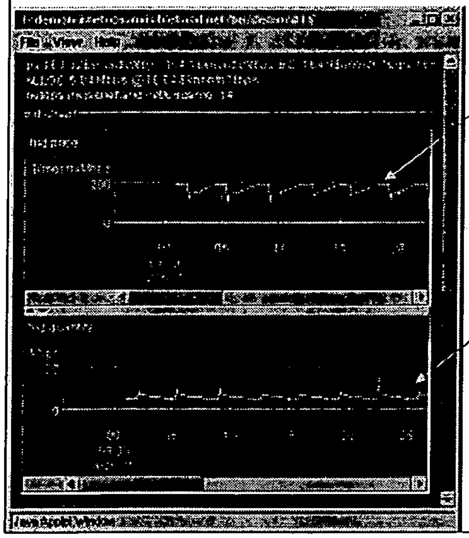
284.25

2.0

Fig 15(m)

"Bid Graph" Subscreen

Bid history over time



Bid price (per unit bandwidth) over time. Cycles correspond to bidding rounds. Graph starts when subscreen is activated.

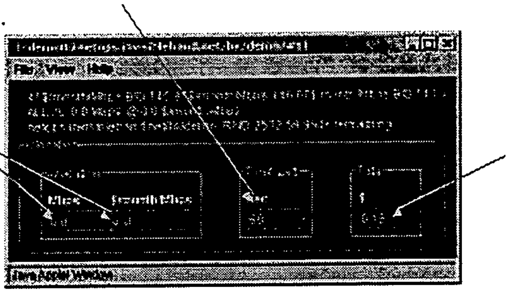
Quantity requested per unit time

Fig 15(n)

"Allocation" subscreen

Results of previous bidding round

If, as is normal, the bidding round is terminated when no bids are received within a configured amount of time, the "Time Left" counter will count down from the configured time, but get reset whenever a bid is received by the Resource Agent, from anyone. When this counter reaches zero, an allocation will be made and a new bidding round will begin after a slight pause to implement the allocation.



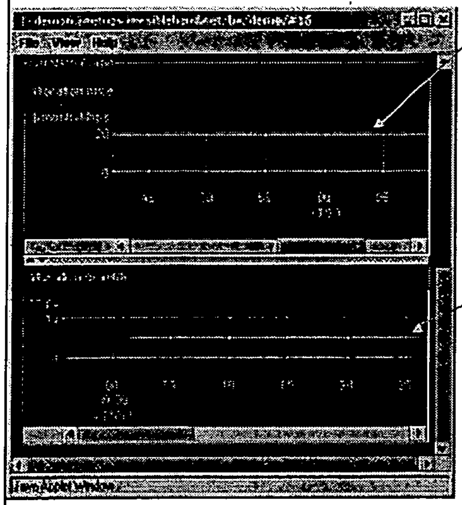
Allocation (Quantity and Price per unit bandwidth) received during the last bid cycle

Total amount spent during this session

Fig 15(o)

“Allocation Graph” Subscreen

Allocation history over time



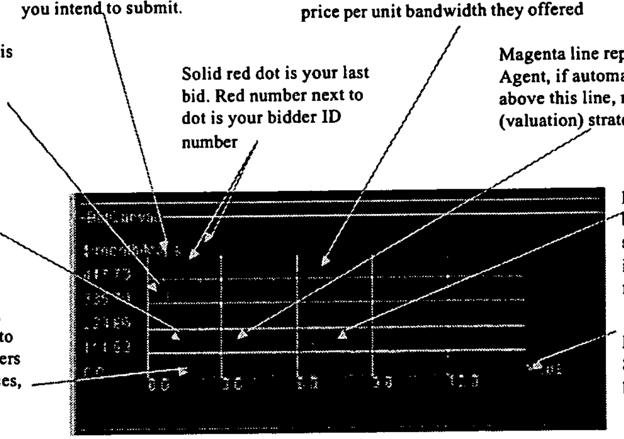
Allocation price (per unit bandwidth) over time. Updated at the end of each bidding round. Graph starts when subscreen is activated.

Quantity received per unit time

Fig 15(p)

“Bid Canvas” Screen

A dynamic display of the second price auction in progress



Red circle is the bid you intend to submit.

Blue dots are other bidders. Their position represents the quantity of bandwidth and the price per unit bandwidth they offered

Solid Magenta dot is the allocation you received last round

Solid red dot is your last bid. Red number next to dot is your bidder ID number

Magenta line represents your budget. Agent, if automated, will never bid above this line, regardless of bid (valuation) strategy line

Magenta circle is the allocation you would get based on your last bid.

Red line represents bidding (valuation) strategy. Your bids, if automated, will follow this line.

Blue shaded area represents bandwidth allocated, from right to left. If different bidders pay different unit prices, it will look like steps. Black space at left indicates that not all bandwidth was allocated based on the current bids.

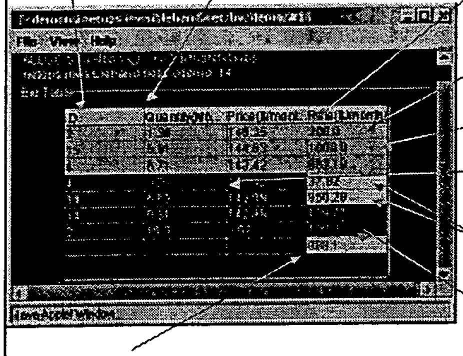
Far right dot is seller. Shows price floor and bandwidth being sold

Note: Manual bids may be placed by placing your cursor at the position which represents your bid and clicking your mouse.

Fig 15(g)

"Bid Table" Subscreen

A dynamic display of the second price auction in progress



ID's of bidders

Columns can be resized by dragging column separators

"Rate" is allocated Quantity times bid Price (per unit bandwidth)

Bidders shaded in blue would receive an allocation of bandwidth if no further bids were received

You are bidder with red text

Bidders with no shading would receive no allocation if all bids remained the same.

Bidders shaded in yellow are those used to calculate the auction price of bandwidth received by the bidder shown in red (you)

This is the "rate" bidder in red (you) would pay for the bandwidth allocated (as opposed to what you bid, above)

Bottom un-shaded bidder is the seller. The seller's "bid" is his price floor

Fig 15(n)